Содержание

1. Лабораторная работа 7. Фиксация событий безопасности средствами системы аудита

1.1. Информация по стенду

- Учетные записи

```
sysadmin:netlab123
root:netlab123
```

- При выполнении заданий лабораторной работы требуется работать под учетной записью суперпользователя. Для перехода в контекст безопасности суперпользователя используйте команду **su -**.

1.2. Системы протоколирования в ОС

- К средствами протоколирования событий в ОС относят
  - **систему журнализации**
  - **систему аудита**

1.2.1. Задача журнализации

Сбор/накопление/обработка событий системы и приложений. Приложения сами решают что отправлять в систему журнализации

- Реализации системы журнализации
  - UNIX-подобные ОС - **syslog**-совместимые службы: rsyslog, syslog-ng
  - Linux - **journald** (часть системы инициализации **systemd**)
  - Windows - EventLog

1.2.2. Задача аудита

Фиксация критичных с точки зрения безопасности системы событий. Администратор определяет, какие события фиксировать.

- Реализации системы аудита
  - Linux - **auditd**
  - Windows - Security Logging

1.3. Системный аудит в ОС Linux

- На основании заданных правил фиксируются **критические с точки зрения безопасности** события в системе

  Подсистема аудита добавлена в ядро Linux начиная с версии 2.6

1.3.1. Примеры фиксируемых событий

- запуск и завершение работы системы
- чтение, запись и изменение прав доступа к файлам
- инициация сетевых соединений
- попытки неудачной авторизации в системе
- изменение сетевых настроек
- изменение информации о пользователях и группах
- запуск и остановка приложений

- выполнение системных вызовов

  Кроме самого факта возникновения события, система аудита представляет такую информацию, как дата и время возникновения события, ответственность пользователя за событие, тип события и его успешность.

1.3.2. Схема работы auditd

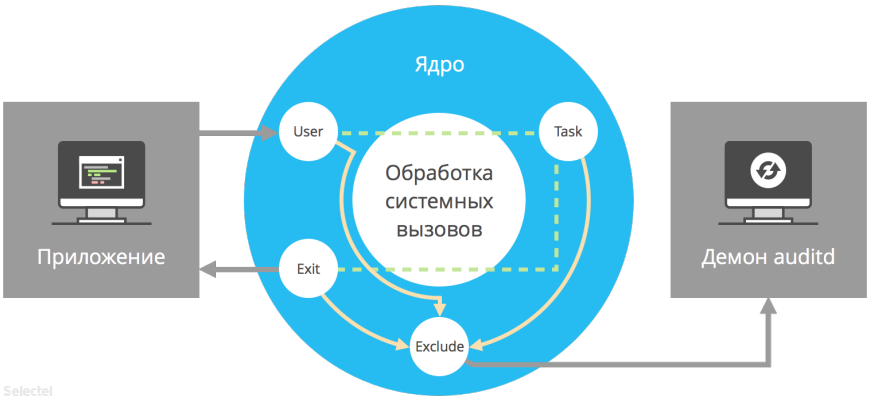

Рис. 1: Схема работы системы аудита

- Ни одно событие в операционной системе не может произойти без использования **системных вызовов** её ядра.

  Запуск нового процесса, открытие файлов и работа с ними, запрос времени ОС Linux, обращение к оборудованию, создание сетевого соединения, вывод информации на экран – все эти операции производятся с помощью обращения к **функциям ядра операционной системы**, для краткости называемых **системными вызовами**.

  Если приложение не использует в своей работе системные вызовы ядра, оно оказывается замкнутым в самом себе и просто **не способно** к какому-либо взаимодействию со своим окружением, не говоря уже о пользователе.

- Для того, чтобы отследить любое **системное событие**, достаточно просто перехватывать все обращения к **системным вызовам**.

  Что и выполняет система аудита.

- Система аудита в ядре ОС устанавливает **триггеры** до и после обработки системных вызовов

Когда происходит системный вызов, триггер срабатывает, подсистема аудита получает всю информацию о вызове и его контексте, передает ее демону **auditd** и отдает дальнейшее управление функции, обрабатывающей системный вызов.

- По срабатыванию триггера система аудита в ядре ОС передает данные о произошедшем событии и его контексте службе **auditd**.

  Событие может быть записано в журнал.

- В процессе работы системы происходит **огромное количество** системных вызовов, поэтому держать все триггеры во включенном состоянии **нецелесообразно**.

  Только во время старта приложение может выполнить несколько сотен-тысяч системных вызовов.

  Отследить выполняемые программой <prog> системные вызовы можно при помощи команды

```
$ strace <prog>
```

- По умолчанию триггеры **отключены** и могут выборочно включаться с помощью **правил аудита**.

  В правилах задается **имя системного вызова**, **вход или выход**, **успешность**, и пр. атрибуты, описывающие **контекст системного вызова**.

- Настраивая **правила аудита** системный администратор может вести наблюдение за любым требуемым аспектом работы операционной системы.

  Поэтому, если в системе происходят какие-либо подозрительные действия, вызванные работой взломщика или вредоносного ПО, с помощью системного аудита не составит труда их выявить, за исключением событий, которые вызваны компонентами непосредственно ядра ОС Linux.

1.3.3. Принцип работы демона auditd

- События записываются в буфер в оперативной памяти
  - ограничивается максимальное число записей (freq)
- Буфер может синхронизироваться с диском
  - немедленно
  - по достижению freq
  - не синхронизироваться
- При дефиците свободного места на диске
  - уведомляет по почте
  - пишет в syslog
  - останавливает аудит
  - выключает узел

- На диске журналы создаются в каталоге
  - **/var/log/audit** в двоичном либо текстовом формате

1.4. Инструменты auditd

- Пакет **audit** - инструменты управления

```
$ apt update && apt install auditd
```

Установка в Debian/Ubuntu Linux

- **auditd** - служба аудита - получает события от ядра по факту срабатывания триггеров, управляемых правилами аудита и записывает их в журнал аудита.

```
$ systemctl start auditd
$ systemctl enable auditd
```

Запуск, автозапуск службы аудита.

- Для того чтобы сделать возможным аудит всех процессов, запущенных до службы аудита, необходимо добавить в строку параметров ядра (в конфигурации загрузчика) параметр **audit=1**.

  В противном случае аудит некоторых процессов будет невозможен.

- **audisp** - диспетчер плагинов - передача событий внешним обработчикам
- **auditctl** - информацию о текущем состоянии подсистемы аудита, добавление и удаление правил
- **autrace** - аудит событий, порождаемых процессами (на подобии strace)
- **ausearch** - поиск событий в журнальных файлах
- **aureport** - генерация отчётов о работе системы аудита
- **/etc/audit/auditd.conf** - настройки подсистемы аудита
- **/etc/audit/audit.rules** - правила аудита, загружаемые при старте
- **/etc/audit/rules.d/** - каталог, содержащий отдельные наборы правил
- **augenrules** - утилита сборки вместе правил аудита

```
$ auditctl -l
```

Рабочий список правил аудита.

1.4.1. Задание 1. Установка инструментов для работы с сообщениями аудита

1. Выполните установку пакета **auditd**.
2. Выполните запуск службы аудита.
3. Настройте автозапуск службы аудита.
4. Ознакомьтесь с рабочим списком правил аудита (скорее всего окажется пустым).

1.5. Настройка сервиса auditd

`/etc/audit/auditd.conf`

Основные параметры службы аудита в конфигурационном файле его службы.

Параметр	Описание
log_file	расположение журналов аудита
log_format	формат журналов
freq	максимальное число записей в буфере
flush	режим синхронизации буфера с диском
num_logs	количество хранимых файлов журнала_
max_log_file	максимальный размер файла в Mb
max_log_file_action	действие при превышении размера
space_left	минимум свободного пространства в Mb
space_left_action	действие при достижении минимума
disk_full_action	действие при переполнении диска

1.5.1. Пример auditd.conf

```
num_logs = 10
max_log_file = 30
max_log_file_action = ROTATE
```

- 10 файлов журналов
- по 30 Мб максимум каждый
- при достижении максимума - ротация

1.5.2. Задание 2. Настройки системы аудита и его журнал

1. Ознакомьтесь с конфигурационном файлом службы аудита.
2. Определите файл с журналом аудита - его расположение задается параметром **log_file** в конфигурационном файле службы.
3. Ознакомьтесь с содержимым журнала аудита.

1.6. Работа с правилами auditd средствами auditctl

Параметры auditctl	Описание
-s	показать статус
-l	вывести список имеющихся правил
-a	добавить новое правило конец списка
-A	добавить новое правило в начало списка
-d	удалить правило из списка

Параметры auditctl	Описание
-D	удалить все имеющиеся правила

Основные параметры утилиты **auditctl**.

1.6.1. Добавление правила аудита (рабочая, временная конфигурация)

- Для задания правил аудита в рабочей (временной) конфигурации используется утилита **auditctl**.

  Конфигурация не будет сохранена и потеряется при перезагрузке узла.

```
$ auditctl -a <список>,<действие> -S <вызов> -F <фильтры>
```

- **Список** - список событий, в который следует добавить правило, всего существует пять списков:
  - **task** - события, связанные с созданием новых процессов
  - **entry** - события при входе в системный вызов
  - **exit** - события при выходе из системного вызова
  - **user** - события, использующие параметры пользовательского пространства, такие как uid, pid и gid
  - **exclude** - используется для исключения ненужных событий
- В большинстве случаев, из них используются только **entry** и **exit**, которые позволяют зарегистрировать либо сам факт обращения к системному вызову, либо его успешную отработку.
- **Действия**:
  - **always** - события будут записываться в журнал
  - **never** - не будут
- Имена системных вызовов задаются параметром **-S**
  - **open** - подразумевается по-умолчанию
  - **close**
  - и т.д.
- **Фильтры** - **-F** - для указания дополнительных параметров фильтрации события
  - **path=<path>** - путь к файлу, каталогу, обращения к которому должны отслеживаться
  - **perm=<perm>** - отслеживаемые методы доступа
    - **a** - изменение атрибутов
    - **r** - чтоение
    - **w** - запись
    - **x** - выполнение

- **loginuid=<UID>** - отслеживать действия пользователя с указанным <UID>
- **arch=[b32|b64]** - отслеживать системные вызовы для определенной архитектуры

В системах с поддержкой нескольких системных архитектур (например 32/64 bit) будет выдаваться предупреждение, если архитектура явно не указана. Архитектуру необходимо указывать **перед** указанием фильтра по системным вызовам.

```
$ auditctl -a exit,always -S open,openat -F path=/etc
```

Регистрировать использование системных вызовов **open()/openat()** при обращении к файлам каталога **/etc**.

```
$ auditctl -a exit,always -F arch=b64 -S open,openat -F path=/etc
```

Регистрировать использование системных вызовов для **64-битной** архитектуры **open()/openat()** при обращении к файлам каталога **/etc**.

```
$ auditctl -a exit,always -F path=/etc -F perm=aw
```

Регистрировать те события, при которых файл из каталога **/etc** открывается на **запись и изменение атрибутов**.

```
$ auditctl -a always,exit -S all -F pid=1005
```

Записывать все системные вызовы, используемые определенным процессом (PID=1005).

```
$ auditctl -a always,exit -S open,openat -F auid=510
```

Записывать все файлы, открытые определенным пользователем (UID=510).

```
$ auditctl -a exit,always -S open,openat -F success!=0
```

Записывать **неудачные попытки** выполнения системного вызова **open()/openat()**.

```
$ auditctl -a always,exit -F path=/etc/shadow -F perm=wa
$ auditctl -w /etc/shadow -p wa
```

Записывать попытки изменения файла **/etc/shadow** (два способа).

1.6.2. Упрощенный синтаксис для аудита файлового I/O

- Для случая, когда система аудита используется для **отслеживания обращения к файлам** на файловой системе подразумевается установка триггера на **выход (exit)** из системных вызовов **open()/openat()**.
- Поскольку это достаточно **типовой сценарий** применения системы аудита в ОС Linux, предусмотрен **упрощенный сисntаксис** - см. ниже.

```
$ auditctl -w /etc/passwd -p war -k password-file
$ auditctl -w /etc/group -p wa -k group_changes
```

- **-w** - какой файл/ каталог отслеживать
- **-p** - какие действия фиксировать
  - **w** - запись
  - **a** - изменение атрибутов
  - **r** - чтение
- **-k** - ключ для дальнейшего поиска данных записей в отчетах **auditd**

1.7. Файлы правил аудита (постоянная конфигурация)

- Постоянная конфигурация правил берется подсистемой аудита из файла:

```
/etc/audit/audit.rules
```

  Указанный файл генерируется на основе файлов правил в каталоге **/etc/audit/rules.d/**.

- Для описания сохраняемой конфигурации аудита вместо использования утилиты **auditctl** стоит выполнять настройку при помощи создания файлов с расширением ***.rules** в указанном каталоге.

- Сохранение текущего списка правил в постоянную конфигурацию

```
$ auditctl -l > /etc/audit/rules.d/my.rules
$ augenrules
$ cat /etc/audit/audit.rules
. . .
```

  Выводим ранее сформированный средствами **auditctl** список правил в файл **my.rules** для создания постоянной конфигурации службы аудита.

  Перегенерируем файл конфигурации службы аудита.

  Убеждаемся, что новые правила добавились в постоянную конфигурацию системы аудита.

- Естественно, можно редактировать правила аудита непосредственно в файлах каталога **/etc/audit/rules.d/**

  После выполнения настроек следует перезапустить службу аудита

```
$ systemctl restart auditd
```

  Далее, стоит убедиться, что добавленные правила активны, посмотрев список активный правил аудита:

```
$ auditctl -l
```

1.7.1. Пример audit.rules

```
# удаляем все ранее созданные правила
-D
# задаём количество буферов, в которых будут храниться сообщения
-b 8192
# при переполнении буферов: 0 - ничего не делать;
# 1 - отправлять сообщение в dmesg/syslog, 2 - отправлять ядро в панику
-f 1
# сами правила
-a exit,always -F path=/etc -F perm=aw
-a exit,always -S open -F loginuid=0

# Наблюдение за конфигурационными файлами
-w /etc/audit/auditd.conf -p wa
-w /etc/audit/audit.rules -p wa

# Наблюдение за журнальными файлами
-w /var/log/audit/
-w /var/log/audit/audit.log

# Настройки наблюдения за заданиями at
-w /etc/at.allow
-w /etc/at.deny
```

1.7.2. Аудит системных вызовов

- Для отслеживания системных вызовов, не связанных с файловым вводом-выводом, необходимо использовать полный, несокращенный синтаксис задания правил аудита.

```
-a <список>,<действие> -S <вызов> -F <фильтры> -k <ключ>
```

1.7.3. Примеры аудита системных вызовов

```
-a always,exit -S adjtimex -S settimeofday -k time_change
-a always,exit -S sethostname -S setdomainname -k system_locale
```

Отслеживание системных вызовов изменения системного времени и установки имени узла-домена с заданием соответствующих ключей поиска.

1.7.4. Задание 3. Добавление правил отслеживания обращения к файлам

1. Выполните добавление правил аудита, отслеживающих обращение на запись к файлам /etc/group, /etc/passwd, /etc/shadow.
2. Задайте для соответствующих событий ключи поиска etc_group, etc_passwd, etc_shadow соответственно.

3. Убедитесь, что правила добавлены и активны.

1.7.5. Задание 4. Выполнение действий, отслеживаемых системой аудита

1. При помощи утилит **useradd**, **groupadd**, **usermod**, **passwd** выполните создание нового пользователя, создание новой группы, добавление созданного пользователя в созданную группу, установку пароля пользователя.
2. Выполните вход в систему под созданным пользователем.

- Выполненные в данном задании действия должны будут зафиксироваться в журнале аудита в соответствии с установленными вами ранее правилами.

1.8. Создание отчетов по логам аудита - aureport

- Лог службы аудита

`/var/log/audit`

Журнальные файлы, создаваемые демоном **auditd** в каталоге **/var/log/audit**, не предназначены для непосредственного чтения, но могут быть прочитаны с помощью специальных утилит, устанавливаемых вместе с самим демоном, основная из которых - утилита **aureport**, генерирующая отчеты из лог-файлов

- Структура отчета - заголовки столбцов в шапке - во всех отчетах, кроме сводки по событиям есть номера событий аудита

`$ ausearch -a <id>`

Получение полных данных о событии

- Сводка по событиям **Summary Report**

`$ aureport`

Вызвав ее без аргументов, мы узнаем общую статистику использования системы, включая такие параметры, как количество входов и выходов из системы, открытых терминалов, системных вызовов и т.д

`$ aureport -f`

Отчет об отслеживаемых файлах и сокетах.

`$ aureport -s`

Сводка по системным вызовам.

`$ aureport -au`

Сводка по событиям аутентификации.

`$ aureport -l`

Отчет о попытках входа в систему.

```
$ aureport -m
```

Отчет об изменениях пользовательских учетных записей.

```
$ aureport  --failed
```

События, завершившиеся неудачно.

```
$ aureport -s --start 08/31/19 12:00 --end now
$ aureport -f --start 03/01/20 12:00 --end 03/20/20 13:00
```

Фильтрация по дате и времени

- Параметры фильтрации по времени
  - **yesterday**
  - **today** - начиная с полуночи
  - **recent** - 10 минут назад
  - **this-week, this-month, this-year**
- Вывод команды разбит на несколько столбцов, которые имеют следующие значения (слева направо):
  - индекс;
  - дата возникновения события;
  - время возникновения события;
  - имя файла;
  - номер системного вызова (чтобы увидеть его имя см. опцию -i);
  - успешность системного вызова (yes или no);
  - имя процесса, вызвавшего событие;
  - audit UID (AUID);
  - номер события.
- Вывод команды **aureport** также можно существенно сократить, если указать флаг **–summary**, который определяет выводить не все случаи доступа к файлом, а только их общее количество по отношению к каждому из файлов:

```
$ aureport -f -i --start recent --summary
```

Вывод команды aureport будет разбит на две колонки, первая из которых отражает количество попыток доступа к файлу (успешных или нет), а вторая – его имя.

- *aureport(8)*

1.8.1. Задание 5. Создание отчета по событиям доступа к файлам.

1. Выполните создание отчета по отслеживаемым файлам.
2. Убедитесь по отчету, что в журнал аудита были записаны события записи в файл **/etc/shadow** при создании пользователя.

3. Запишите содержимое отчета в файл **shadow.report**

4. Полученный файл **shadow.report** является одним из результатов выполнения работы.

1.9. Поиск и анализ событий - ausearch

- Инструмент поиска по событиям аудита

  Принимает параметры для поиска. Все условия, заданные в параметрах объединяются по логическому **И**

```
$ ausearch -a <event_id>
```

- Поиск события по номеру (Event ID). Номера можно выяснить из вывода aureport, кроме сводного отчета

```
$ ausearch -ui 1001 -i
```

  Детальный отчет обо всех действиях пользователя с UID=1001

- **-i**

  преобразование числовых данных в символьные

```
$ ausearch -f /etc/passwd -i
```

  Отчет о доступе к файлу **/etc/passwd**

```
$ ausearch -k password-file -ui 1001
```

  Поиск по дополнительному ключу, задаваемому в правиле

```
$ ausearch -sc ptrace -i
```

  по именам системных вызовов

```
$ ausearch -x /usr/bin/nmap
```

  по именам исполняемых файлов

```
$ ausearch -tm pts/0
```

  по имени терминала

```
$ ausearch -tm cron
```

  по именам демонов

- Вывод **ausearch** также может быть сокращен с помощью указания временных периодов, которые использовались при вызове **aureport**.

- *ausearch(8)*

1.9.1. Примеры поиска событий

- Поиск записей аудита, связанных с использованием механизма аутентификации

```
$ ausearch -m USER_AUTH
```

- Поиск записей аудита, связанных с использованием механизма идентификации

```
$ ausearch -m USER_LOGIN –i
```

1.9.2. Задание 6. Поиск события в журнале аудита

1. По заданном в задании 3 ключу для обращения к файлу с паролями пользователей (/etc/shadow) выполните средствами утилиты **ausearch** поиск в журнале аудита событий с данным ключем. Результат поиска должен содержать полное описание события - установки пароля пользователя утилитой **passwd**.
2. Запишите результат поиска в файл **shadow.search**
3. Полученный файл **shadow.search** является одним из результатов выполнения работы.

1.10. Итоговое задание. Загрузка результата выполнения лабораторной работы

- В качестве результата выполнения лабораторной работы необходимо загрузить в Moodle следующие файлы:
  - Файл **shadow.report**, полученный в задании 5
  - Файл **shadow.search**, полученный в задании 6